DETAILED ACTION

Information Disclosure Statement

1. The information disclosure statements (IDSs) submitted on 4/29/2009, 11/13/2009, 1/20/2010, 3/17/2010 and 9/7/2011 are in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statements are being considered by the examiner.

EXAMINER'S AMENDMENT

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Stanley C. Spooner (Reg. No. 27,393) on December 21, 2011.

3. The Application has been amended as follow:

In the claims:

Claim 1 (Currently Amended). A method of controlling execution of a processing task within a data processing system, said method comprising the steps of:

executing said processing task including allocating memory areas for data storage; and

suspending an actual execution path of said processing task at an execution point to perform memory management, said memory management comprising the steps of:

identifying at least one data item roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

determining a correlation between reference values corresponding to said at least one data item roots and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item roots; [[and]]

performing a memory management operation on allocated memory areas in dependence upon said correlation;

wherein said identifying step comprises:

identifying a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path; performing a simulated execution of said possible execution path; and

wherein said at least one data item roots and said at least one data item accessible to said processing task are identified by following said possible execution path to said current execution point.

Claim 3 (Canceled).

Claim 4 (Currently Amended). A method as claimed in claim [[3]] 1, wherein said memory management operation comprises marking all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and collecting unmarked memory areas for re-allocation during subsequent execution of said processing task.

Claim 7 (Currently Amended). A method as claimed in claim 6, wherein said identifying step comprises:

scanning a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data items;

[[categorising]] categorizing at least one of said data item roots or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

simulating all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item [[categorised]] categorized as a multiple-type variable and determining the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

checking said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

said memory management operation is performed in dependence upon a result of said step of checking said determined data type.

Claim 11 (Currently Amended). A computer program product comprising a non-transitory computer readable storage medium comprising computer readable instructions that, when executed, cause a computer to control execution of a processing task within a data processing system, said computer program comprising:

execution code operable to execute said processing task including allocating memory areas for data storage; and

suspending code operable to suspend an actual execution path of said processing task at an execution point to perform memory management;

reference identifying code operable to identify at least one data item roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlating code operable to determine a correlation between reference values corresponding to said identified data item root and memory areas allocated during said execution up to said execution point; [[and]]

memory management code operable to perform a memory management operation on allocated memory areas in dependence upon said correlation;

wherein said reference identifying code comprises:

path identifying code operable to identifying a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

path simulation code operable to perform a simulated execution of said possible execution path; and

wherein said at least one data item root and said at least one data item accessible to said processing task are identified by following said possible execution path to said current execution point.

Claim 13 (Canceled).

Claim 14 (Currently Amended). A computer program product as claimed in claim [[13]] 11, wherein said memory management code is operable to mark all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and to collect [[umarked]] unmarked memory areas for re-allocation during subsequent execution of said processing task.

Claim 17 (Currently Amended). A computer program product as claimed in claim 16, wherein said reference identifying code comprises:

scanning code operable to scan a plurality of program instructions corresponding to said processing task and to log a data type for each store instruction corresponding to each of said one or more data items;

[[categorising]] categorizing code operable to [[categorise]] categorize at least one of said one or more data items as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item [[categorised]] categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code.

Claim 21 (Currently Amended). A data processing apparatus operable to control execution of a processing task within a data processing system, said data processing apparatus comprising:

a processor and a memory;

execution logic operable to execute said processing task including allocating memory areas for data storage; and

task suspension logic operable to suspend an actual execution path of said processing task at an execution point to perform memory management;

reference identifying logic operable to identify at least one data item occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlation logic operable to determine a correlation between reference values corresponding to identified said at least one data item root and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item root; [[and]]

memory management logic operable to perform a memory management operation on allocated memory areas in dependence upon said correlation;

wherein said reference identifying logic comprises:

path identifying logic operable to identify a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

path simulation logic operable to perform a simulated execution of said possible execution path; and

wherein said at least one data item and said at least one data item root accessible to said processing task are identified by following said possible execution path to said current execution point.

Claim 23 (Canceled).

Claim 24 (Currently Amended). A data processing apparatus as claimed in claim [[23]] 21, wherein said memory management logic is operable to mark all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and collecting [[umarked]] unmarked memory areas for re-allocation during subsequent execution of said processing task.

Claim 27 (Currently Amended). A data processing apparatus as claimed in claim 26, wherein said reference identifying logic comprises:

scanning logic operable to scan a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said one or more data items;

[[categorising]] categorizing logic operable to [[categorise]] categorize at least one of said at least one data item roots or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate all possible execution paths up to said execution point for each of said at least one data [[itemroot]] item root or said at least one data item [[categorised]] categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

said wherein memory management logic is operable to perform said memory management operation in dependence upon a result of checking said determined data type.

Claim 32 (Currently Amended). A computer program product comprising a non-transitory computer readable storage medium comprising computer readable instructions that, when executed, control a computer to identify for a memory management operation performed by memory management code at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning code operable to scan a plurality of program instructions corresponding to said processing task and to log a data type for each store instruction corresponding to each of said at least one data item;

categorizing code operable to categorize at least one of said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with

each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code.

Claim 33 (Currently Amended). A data processing apparatus for identifying, for a memory management operation performed by memory management logic, at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

a processor and a memory;

scanning logic operable to scan a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data item;

categorizing logic operable to categorize at least one of said at least one data item root or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with

each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management logic is operable to perform said memory management operation in dependence upon a result of checking said determined data type.

Allowable Subject Matter

4. Claims 1, 2, 4-12, 14-22 and 24-33 are allowed.

Conclusion

5. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Darnell (US 2003/0177152), Bush et al. (US 2002/0052926) and LI (US 2004/0039758) all teach a method and a system of garbage collection during the execution of the thread/task.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kaushikkumar Patel whose telephone number is (571)272-5536. The examiner can normally be reached on 8.00 am - 5.00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Matt Kim can be reached on (571) 272-4182. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Kaushik Patel/
Kaushikkumar Patel
Primary Examiner
Art Unit 2186